O'REILLY®

"This is a new generation of CSS books, for a new generation of CSS. Nobody is better at making sense of this new C Lea Verou—among the handful of truly amazing coders I've
—Jeffrey Zeldman, author, *Designing With Web St*

CSS SECRETS

BETTER SOLUTIONS TO EVERYDAY WEB DESIGN PROBLEMS

LEA VEROU

FOREWORD BY ERIC A. MEYER

18 Frosted glass effect

Prerequisites

RGBA/HSLA colors

The problem

We are using the term "backdrop" here to mean **the part of the page that is underneath an element**, which shows through its semi-transparent background.

One of the first use cases of semi-transparent colors was using them as backgrounds, over photographic or otherwise busy backdrops, to decrease contrast and make the text possible to read. The result is quite impressive, but can still be hard to read, especially with very low opacity colors and/or busy backdrops. For example, take a look at **Figure 4.14**, where the main element has a semi-transparent white background. The markup looks like this:

HTML

```
<main>
    <blockquote>
        “The only way to get rid of a temptation[…]”
        <footer>—
            <cite>
                Oscar Wilde,
                The Picture of Dorian Gray
            </cite>
        </footer>
    </blockquote>
</main>
```

And the CSS looks like this (with all irrelevant bits omitted for brevity):

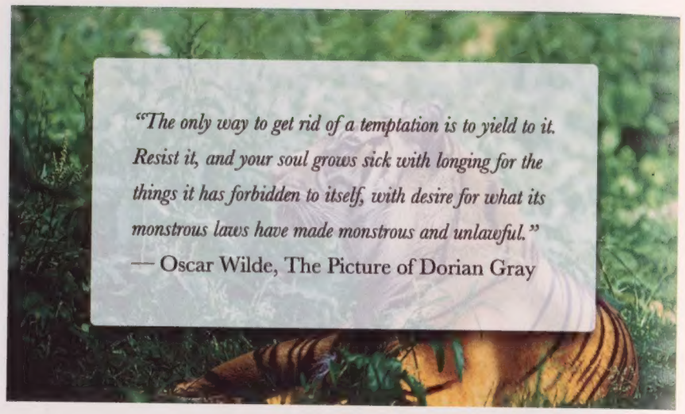
```
body {
    background: url("tiger.jpg") 0 / cover fixed;
}

main {
    background: hsla(0,0%,100%,.3);
}
```

As you can observe, the text is really hard to read, due to the image behind it being busy and the background color only being 25% opaque. We could

FIGURE 4.14
Our semi-transparent white background makes the text hard to read

FIGURE 4.15
Increasing the alpha value of our background color does fix the readability issue, but also makes our design less interesting



of course improve readability by increasing the alpha parameter of the background color, but then the effect will not be as interesting (see **Figure 4.15**).

In traditional print design, this issue is often addressed by **blurring the part of the photo that is underneath our text container**. Blurred backgrounds are not as busy, and thus, text on them is easier to read. Because blurring is computationally expensive, in the past its toll on resources was prohibitive for using this technique in websites and UI design. However, with GPUs improving and hardware acceleration becoming more commonplace for more and more things, these days it's used quite frequently. In the past few years, we have seen this technique in newer versions of both Microsoft Windows, as well as Apple iOS and Mac OS X (**Figure 4.16**).

FIGURE 4.16
Translucent UIs with a blurred backdrop have been becoming increasingly common in the past few years, as the toll of blurring on resources has stopped being prohibitively expensive (**Apple iOS 8.1** is shown on the left and **Apple OS X Yosemite** is shown on the right)

We also got the ability to blur elements in CSS, via the `blur()` filter, which is essentially a hardware-accelerated version of the corresponding SVG blur filter primitive that we always had for SVG elements. However, if we directly apply a `blur()` filter to our example, the entire element is blurred, which makes it even less readable. (**Figure 4.17**). Is there any way to just apply it to the element's backdrop (i.e., the part of the background that is **behind** our element)?

FIGURE 4.17
Applying a `blur()` filter to the element itself makes things worse

The solution

Provided that our element has a `background-attachment` of `fixed`, this is possible, albeit a bit tricky. Because we cannot apply the blurring to our element itself, **we will apply it to a pseudo-element that is positioned behind the element and whose background seamlessly matches the one on `<body>`**.

It's also possible even with non-fixed backgrounds, just messier.

First, we add the pseudo-element and position it absolutely, with all offsets being `0`, so that it covers the entire `<main>` element:

```
main {
    position: relative;
    /* [Rest of styling] */
```

```
}

main::before {
    content: '';
    position: absolute;
    top: 0; right: 0; bottom: 0; left: 0;
    background: rgba(255,0,0,.5); /* for debugging */
}
```

Be careful when using a negative **z-index** to move a child underneath its parent: if said parent is nested within other elements with backgrounds, the child will go below those as well.

We also applied a semi-transparent **red** background, so we can see what we're doing, otherwise debugging becomes difficult when we're dealing with a transparent (and therefore, invisible) element. As you can see in **Figure 4.18**, our pseudo-element is currently **above** our content, thus obscuring it. We can fix this by adding **z-index: -1;** (**Figure 4.20**).

Now it's time to replace that semi-transparent red background, with one that actually matches our backdrop, either by copying over the **<body>** background, or by splitting it into its own rule. Can we blur now? Let's try it:

Why not just use **background: inherit** on **main::before**? Because then it will inherit from **main**, not **body**, so the pseudo-element will get a semi-transparent white background as well.

```
body, main::before {
    background: url("tiger.jpg") 0 / cover fixed;
}

main {
    position: relative;
    background: hsla(0,0%,100%,.3);
}

main::before {
    content: '';
    position: absolute;
    top: 0; right: 0; bottom: 0; left: 0;
    filter: blur(20px);
}
```


FIGURE 4.18
The pseudo-element is currently obscuring the text

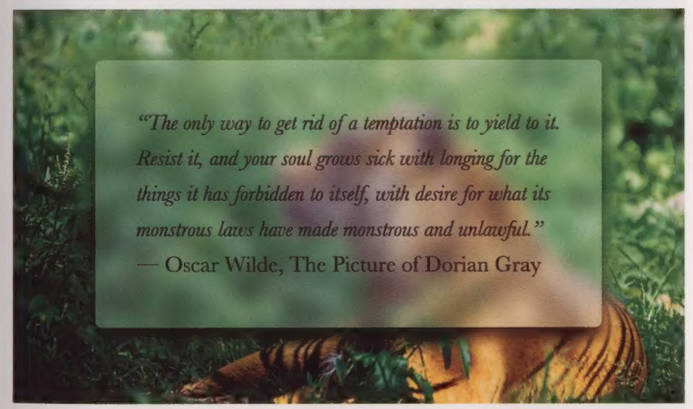
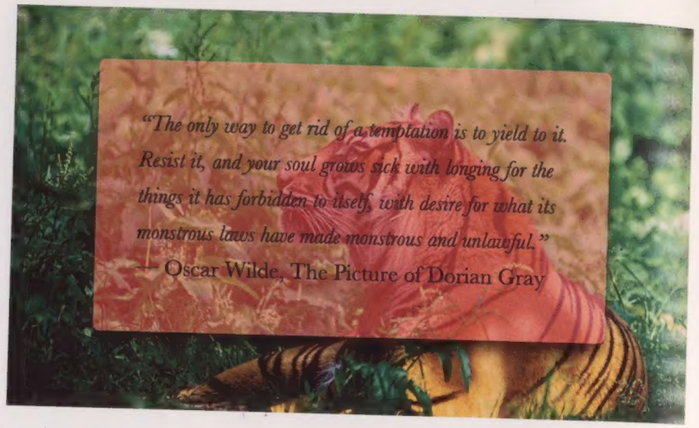


FIGURE 4.19
We fixed the faded blurring at the edges, but now there is some blurring outside our element too

As you can see in **Figure 4.21**, we're pretty much there. The blurring effect looks perfect toward the middle, but is less blurred closer to the edges. This happens because blurring reduces the area that is covered by a solid color by the blur radius. Applying a **red** background to our pseudo-element helps clarify what's going on (**Figure 4.22**).

FIGURE 4.20
Moving the pseudo-element behind its parent, with `z-index: -1;`



To circumvent this issue, we will make the pseudo-element **at least** `20px` (as much as our blur radius) **larger than the dimensions of its container**, by applying a margin of `-20px` or less to be on the safe side, as different browsers might use different blurring algorithms. As **Figure 4.19** demonstrates, this fixes the issue with the faded blurring at the edges, but now there is also **some blurring outside** our container, which makes it look like a smudge instead of frosted glass. Thankfully, this is also easy to fix: we will just apply `overflow: hidden;` to `main`, in order to clip that extraneous blurring. The final code looks as follows, and its result can be seen in **Figure 4.23**:

```
body, main::before {
    background: url("tiger.jpg") 0 / cover fixed;
}

main {
    position: relative;
    background: hsla(0,0%,100%,.3);
    overflow: hidden;
}

main::before {
    content: '';
    position: absolute;
    top: 0; right: 0; bottom: 0; left: 0;
    filter: blur(20px);
    margin: -30px;
}
```

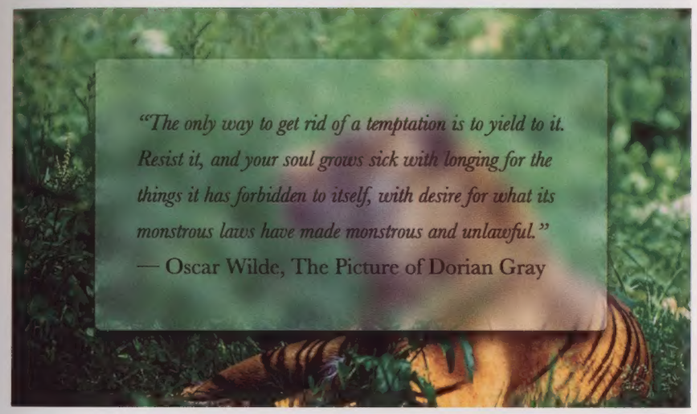


FIGURE 4.21
Blurring our pseudo-element almost works, but its less blurry on the edges, diminishing the frosted glass illusion

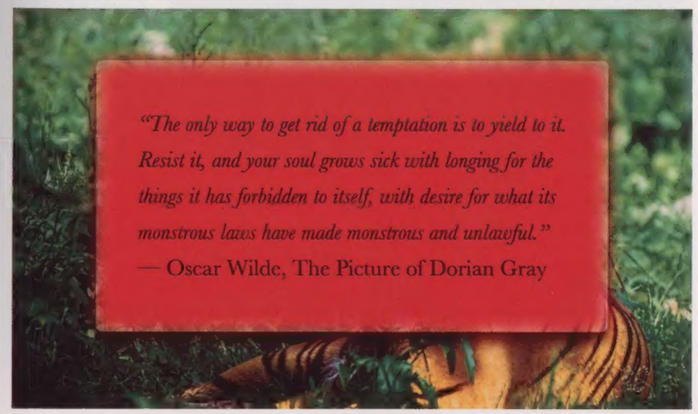
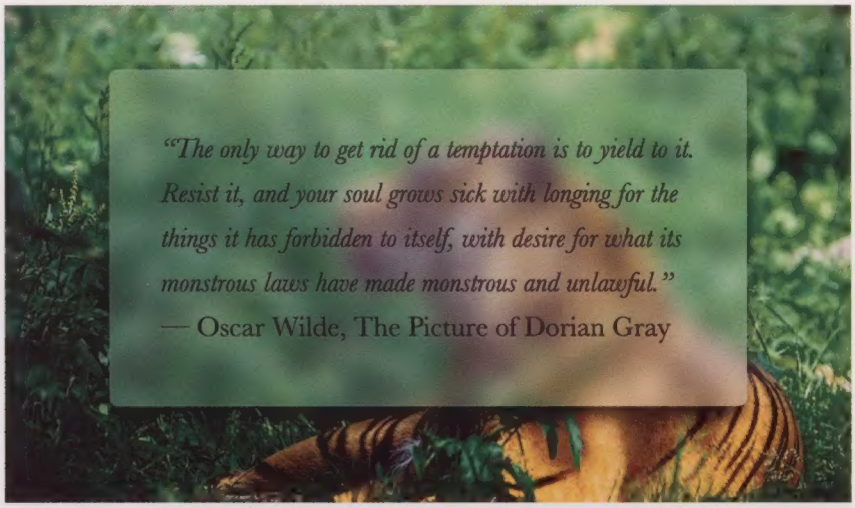


FIGURE 4.22
Adding a `red` background helps make sense of what's happening

FIGURE 4.23
Our final result



Note how much more readable our page has now become, and how much more elegant it looks. It's debatable whether the fallback for this effect constitutes graceful degradation. If filters are not supported, we will get the result we saw in the beginning (**Figure 4.14**). We can make our fallback a bit more readable by increasing the opacity of the background color.

▸ **PLAY!** `play.csssecrets.io/frosted-glass`

- **Filter Effects**
  w3.org/TR/filter-effects

RELATED SPECS

CSS SECRETS

BETTER SOLUTIONS TO EVERYDAY WEB DESIGN PROBLEMS

In this practical guide, CSS expert Lea Verou provides 47 undocumented techniques and tips to help intermediate-to-advanced CSS developers devise elegant solutions to a wide range of everyday web design problems.

Rather than focus on design, *CSS Secrets* shows you how to solve problems with code. You'll learn how to apply Lea's analytical approach to practically every CSS problem you face to attain DRY, maintainable, flexible, lightweight, and standards-compliant results.

Inspired by her popular talks at over 60 international web development conferences, Lea Verou provides a wealth of information for topics including:

- **Background & Borders**
- **Shapes**
- **Visual Effects**
- **Typography**
- **User Experience**
- **Structure & Layout**
- **Transitions & Animations**

CSS/Web Development

US $39.99 CAN $45.99

ISBN: 978-1-449-37263-7

9 781449 372637 53999

"Lea Verou's encyclopaedic mind is one of a kind, but thanks to this generous book, you too can get an insight into what it's like to wield CSS to do just about anything you can think of. Even if you think you know CSS inside-out, I guarantee that there are still secrets in this book waiting to be revealed."

—Jeremy Keith
Shepherd of Unknown Futures, Clearleft

"If you want the inside scoop on fascinating CSS techniques, smart best practices, and some flat-out brilliance, don't hesitate—read this book. I loved it!"

—Eric A. Meyer

"*CSS Secrets* is an instant classic—so many wonderful tips and tricks you can use right away to enhance your UX designs!"

—Christopher Schmitt
Author of *CSS Cookbook*

"Lea is an exceedingly clever coder. This book is absolutely packed with clever and useful ideas, even for people who know CSS well. Even better, you'll feel more clever in your work as this book encourages pushing beyond the obvious."

—Chris Coyier
CodePen

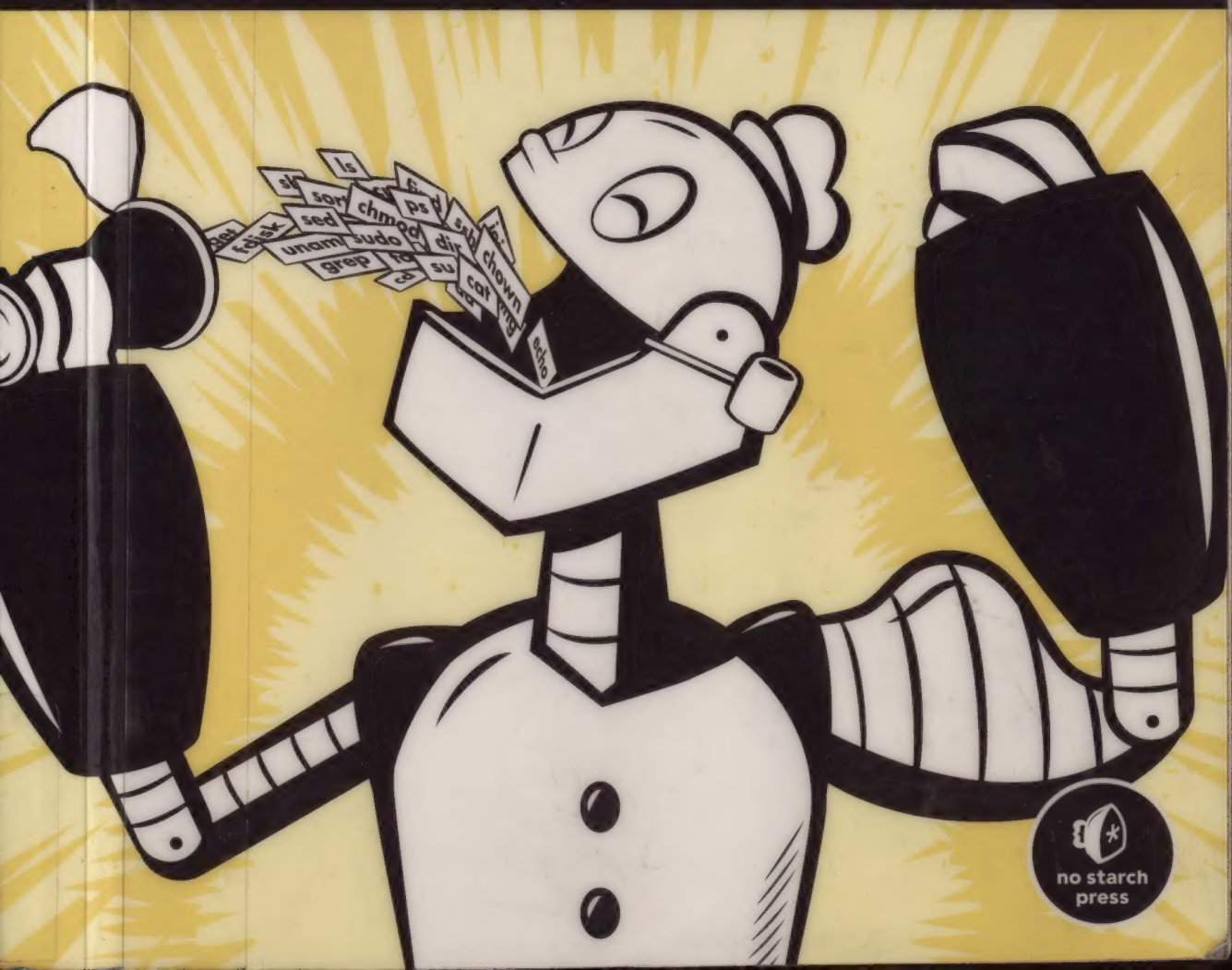
O'REILLY®
oreilly.com

2ND EDIT

THE LINUX COMMAND LINE

A *COMPLETE* INTRODUCTION

WILLIAM SHOTTS

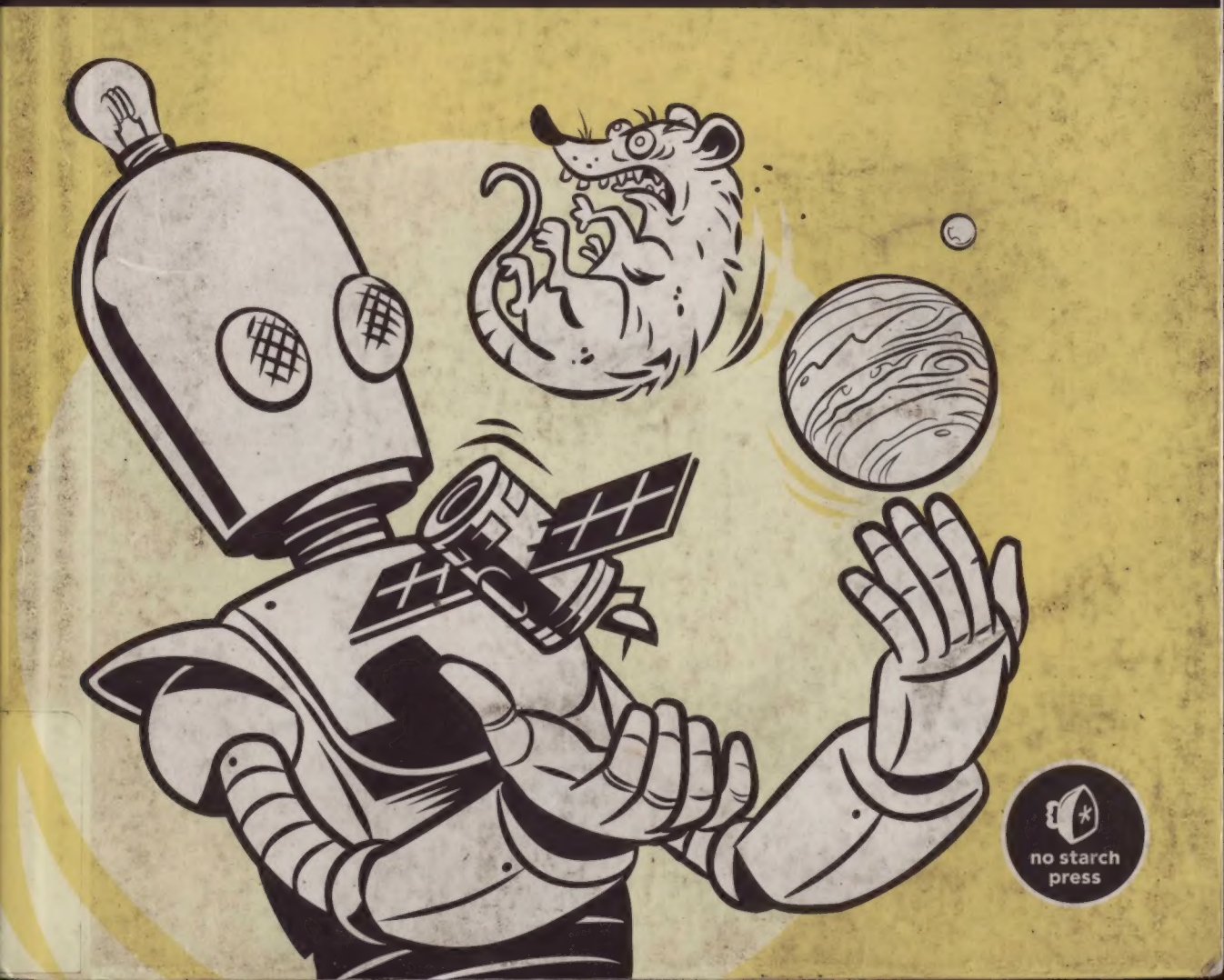


no starch press

IMPRACTICAL PYTHON PROJECTS

PLAYFUL PROGRAMMING ACTIVITIES TO MAKE YOU SMARTER

LEE VAUGHAN



no starch press

CSS SECRETS

BETTER SOLUTIONS TO EVERYDAY WEB DESIGN PROBLEMS

VEROU

2ND EDITION

THE LINUX COMMAND LINE

SHOTTS

IMPRACTICAL PYTHON PROJECTS

VAUGHAN

HTML5 Cookbook

Schmitt
Simpso

HACKS

HTML5 Hacks

Cravens & Burtoft

O'RE

THE NEW WEB TYPOGRAPHY

BOSS & CRANFORD TEA

Shell Programming in Unix, Linux and OS X

Fourth Edition

Kochan

HTML5 Developer's Cookbook

Addison Wesley

HTML5

MacDonald

Second

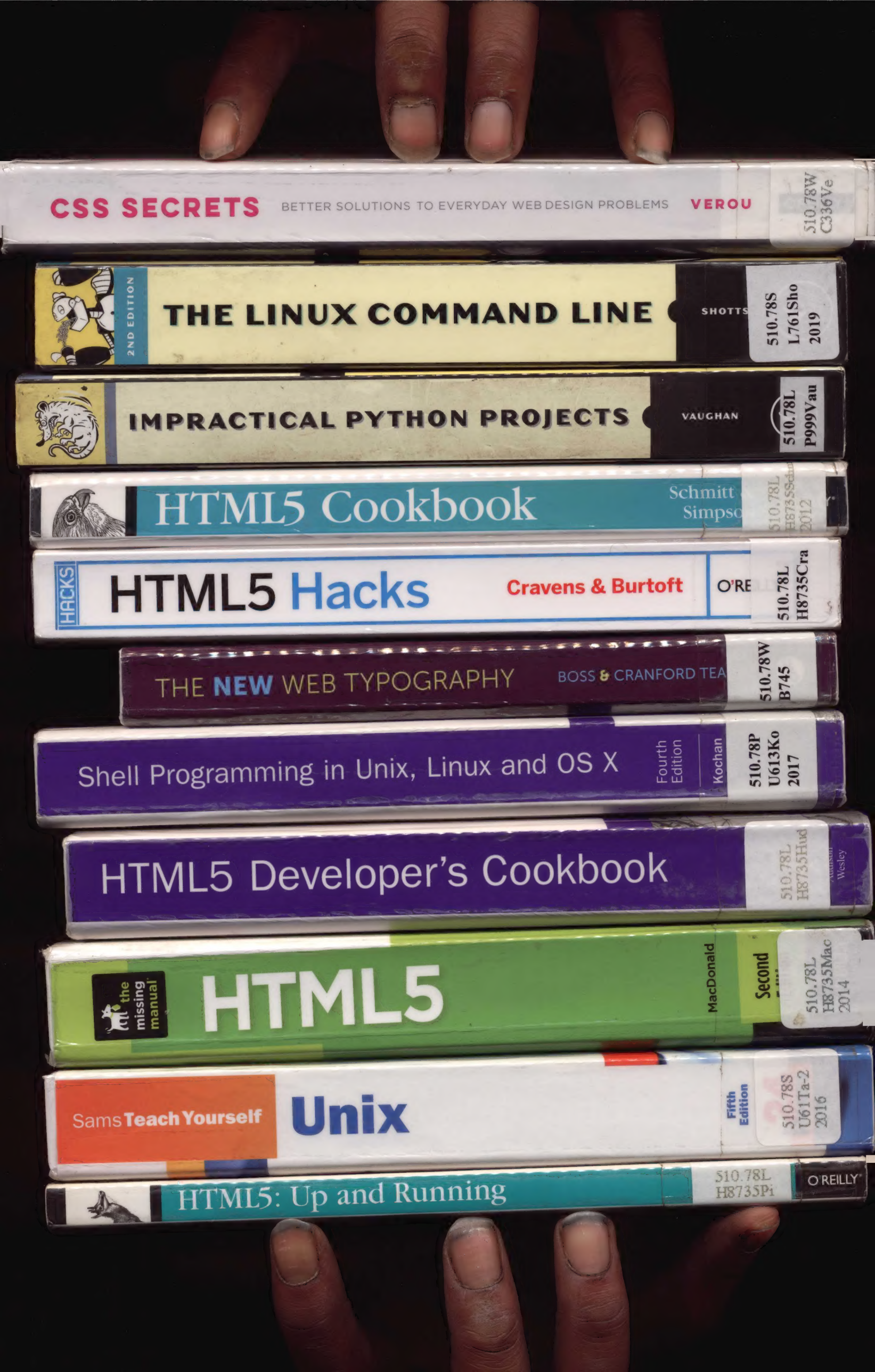
Sams Teach Yourself

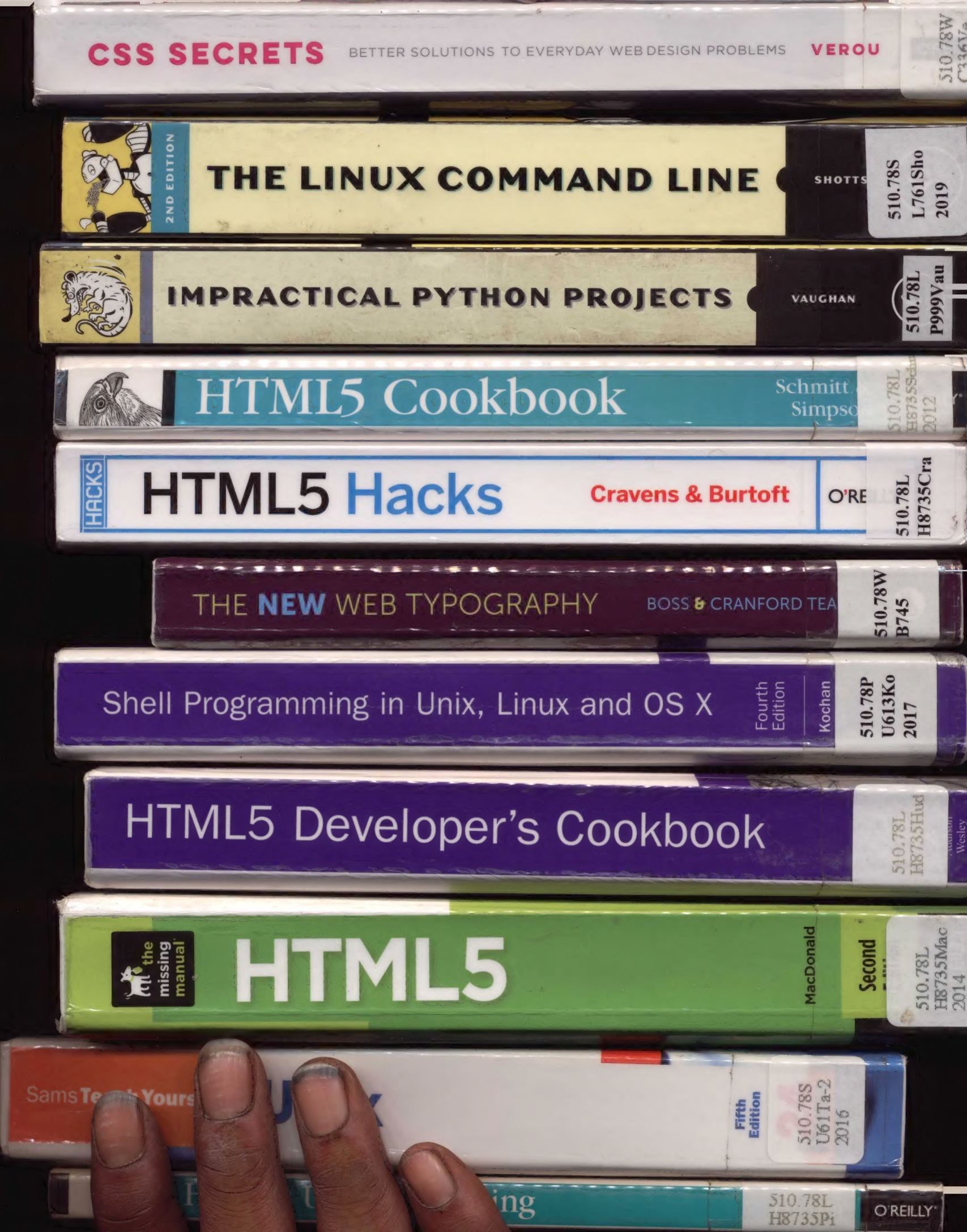
Unix

Fifth Edition

HTML5: Up and Running

O'REILLY


CSS SECRETS
BETTER SOLUTIONS TO EVERYDAY WEB DESIGN PROBLEMS
VEROU
510.78W C336Ve
2ND EDITION
THE LINUX COMMAND LINE
SHOTTS
510.78S L761Sho 2019
IMPRACTICAL PYTHON PROJECTS
VAUGHAN
510.78L P999Vau
HTML5 Cookbook
Schmitt
Simpso
510.78L
2012
HACKS
HTML5 Hacks
Cravens & Burtoft
O'RE
510.78L H8735Cra
THE NEW WEB TYPOGRAPHY
BOSS & CRANFORD TEA
510.78W B745
Shell Programming in Unix, Linux and OS X
Fourth Edition
Kochan
510.78P U613Ko 2017
HTML5 Developer's Cookbook
510.78L H8735Hud
Wesley
the missing manual
HTML5
MacDonald
Second
510.78L H8735Mac 2014
Sams Teach Yourself
Unix
Fifth Edition
510.78S U61Ta-2 2016
HTML5: Up and Running
510.78L H8735Pi
O'REILLY


CSS SECRETS
BETTER SOLUTIONS TO EVERYDAY WEB DESIGN PROBLEMS
VEROU
510.78W
2ND EDITION
THE LINUX COMMAND LINE
SHOTTS
510.78S
L761Sho
2019
IMPRACTICAL PYTHON PROJECTS
VAUGHAN
510.78L
P999Vau
HTML5 Cookbook
Schmitt
Simpso
510.78L
2012
HACKS
HTML5 Hacks
Cravens & Burtoft
O'RE
510.78L
H8735Cra
THE NEW WEB TYPOGRAPHY
BOSS & CRANFORD TEA
510.78W
B745
Shell Programming in Unix, Linux and OS X
Fourth Edition
Kochan
510.78P
U613Ko
2017
HTML5 Developer's Cookbook
510.78L
H8735Hud
the missing manual
HTML5
MacDonald
Second
510.78L
H8735Mac
2014
Sams
Yours
Fifth Edition
510.78S
U61Ta-2
2016
510.78L
H8735Pi
O'REILLY